

**OSD Skill-10****Procedure to add shared memory IPC to xv6**

Step 1:

git clone <https://github.com/hayleejane3/shared-memory-xv6>

Step 2:

cd shared-memory-xv6

Step 3:

ls

Step 4:

vi Makefile

Delete following code from Makefile line 54 to 72

# If the makefile can't find QEMU, specify its path here

#QEMU :=

# Try to infer the correct QEMU if not specified

ifndef QEMU

QEMU := \$(shell if which qemu 1> /dev/null 2> /dev/null; \

then echo qemu; exit; \

else \

qemu=/u/c/s/cs537-2/ta/tools/qemu; \

if test -x \$\$qemu; then echo \$\$qemu; exit; fi; fi; \

echo "\*\*\*\*" 1>&2; \

echo "\*\*\*\* Error: Couldn't find a working QEMU executable." 1>&2; \

echo "\*\*\*\* Is the directory containing the qemu binary in your " 1>&2; \

echo "\*\*\*\* PATH or have you tried setting the QEMU variable in " 1>&2; \

echo "\*\*\*\* Makefile?" 1>&2; \

echo "\*\*\*\*" 1>&2; exit 1)

endif

Add the following code inplace of above code

# QEMU = qemu-system-i386

```
# Try to infer the correct QEMU
ifndef QEMU
QEMU = $(shell if which qemu > /dev/null; \
    then echo qemu; exit; \
    elif which qemu-system-i386 > /dev/null; \
    then echo qemu-system-i386; exit; \
    elif which qemu-system-x86_64 > /dev/null; \
    then echo qemu-system-x86_64; exit; \
    else \
    qemu=/Applications/Q.app/Contents/MacOS/i386-softmmu.app/Contents/MacOS/i386-softmmu; \
    if test -x $$qemu; then echo $$qemu; exit; fi; fi; \
    echo "****" 1>&2; \
    echo "**** Error: Couldn't find a working QEMU executable." 1>&2; \
    echo "**** Is the directory containing the qemu binary in your PATH"
1>&2; \
    echo "**** or have you tried setting the QEMU variable in Makefile?"
1>&2; \
    echo "****" 1>&2; exit 1)
endif
```

```
# QEMU = qemu-system-i386

# Try to infer the correct QEMU
ifndef QEMU
QEMU = $(shell if which qemu > /dev/null; \
    then echo qemu; exit; \
    elif which qemu-system-i386 > /dev/null; \
    then echo qemu-system-i386; exit; \
    elif which qemu-system-x86_64 > /dev/null; \
    then echo qemu-system-x86_64; exit; \
    else \
    qemu=/Applications/Q.app/Contents/MacOS/i386-softmmu.app/Contents/MacOS/i386-softmmu; \
    if test -x $$qemu; then echo $$qemu; exit; fi; fi; \
    echo "****" 1>&2; \
    echo "**** Error: Couldn't find a working QEMU executable." 1>&2; \
    echo "**** Is the directory containing the qemu binary in your PATH" 1>&2; \
    echo "**** or have you tried setting the QEMU variable in Makefile?" 1>&2; \
    echo "****" 1>&2; exit 1)
endif
```

Step 5:  
\$ cd user

Step 6:  
\$ls

Step 7:

## vi test\_shm.c

```
#include "types.h"
#include "stat.h"
#include "user.h"

#define SHM_SIZE 4096U
#define MAX_MEM 0xA0000U

void
create_new_proc(void (*testcase)(void)) {
    int pid = fork();
    int i;
    if (pid < 0) {
        printf(2, "test_shm : create_new_proc failed.\n");
    } else if (pid == 0) {
        testcase();
        exit();
    } else {
        if (wait() != pid) {
            printf(2, "test_shm : create_new_proc wait %d failed\n", pid);
        }
        for (i = 0; i < 8; i++) {
            if (shm_refcount(i) != 0) {
                printf(2, "test_shm : wrong shm count for key %d = %d, not 0\n", i, shm_refcount(i));
            }
        }
    }
}

// no enough mem for heap
void
test_single_proc(void)
{
    int i;
    int sum = 0;
    int *a, *b, *c, *d, *e, *f, *g, *h;
    int *a2;
    uint stack_loc, free_space;
    char *h1, *h2; // heap
    printf(1, "test_single_proc\n");
    a = (int*)shmgetat(0, 1);
    int pgint = SHM_SIZE / sizeof(int);
    printf(1, "shmgetat succeeds, a=%x\n", a);
    if ((uint)a != MAX_MEM - SHM_SIZE) {
        printf(2, "test single proc: failed. address of a: %u, expected address: %u\n",
            a, MAX_MEM - SHM_SIZE);
    }
    for (i = 0; i < pgint; i++) {
        a[i] = i;
    }
    for (i = 0; i < pgint; i++) {
        sum += a[i];
    }
    if (sum != pgint) {
        printf(2, "test single proc: failed. sum=%d, expected=%d\n",
            sum, pgint);
    }
    b = (int*)shmgetat(1, 6); // invalid page number
    if ((int)b != -1) {
        printf(2, "test single proc: b should be -1 due to invalid page number");
    }
    b = (int*)shmgetat(-1, 2); // invalid key
    if ((int)b != -1) {
        printf(2, "test single proc: b should be -1 due to invalid key");
    }
}
```

```

b = (int*)shmgetat(1, 4);
if ((uint)b != (uint)a - SHM_SIZE * 4) {
    printf(2, "test single proc: failed. address of b: %x, expected address: %x\n",
        b, (uint)a - SHM_SIZE * 4);
}
c = (int*)shmgetat(2, 3);
if ((uint)c != (uint)b - SHM_SIZE * 3) {
    printf(2, "test single proc: failed. address of c: %x, expected address: %x\n",
        c, (uint)b - SHM_SIZE * 3);
}
d = (int*)shmgetat(3, 2);
if ((uint)d != (uint)c - SHM_SIZE * 2) {
    printf(2, "test single proc: failed. address of d: %x, expected address: %x\n",
        d, (uint)c - SHM_SIZE * 2);
}
e = (int*)shmgetat(4, 1);
if ((uint)e != (uint)d - SHM_SIZE) {
    printf(2, "test single proc: failed. address of e: %x, expected address: %x\n",
        e, (uint)d - SHM_SIZE);
}
f = (int*)shmgetat(5, 1);
if ((uint)f != (uint)e - SHM_SIZE) {
    printf(2, "test single proc: failed. address of f: %x, expected address: %x\n",
        f, (uint)e - SHM_SIZE);
}
g = (int*)shmgetat(6, 1);
if ((uint)g != (uint)f - SHM_SIZE) {
    printf(2, "test single proc: failed. address of g: %x, expected address: %x\n",
        g, (uint)f - SHM_SIZE);
}
h = (int*)shmgetat(7, 1);
if ((uint)h != (uint)g - SHM_SIZE) {
    printf(2, "test single proc: failed. address of h: %x, expected address: %x\n",
        h, (uint)g - SHM_SIZE);
}

// get key 0 twice
a2 = (int*)shmgetat(0, 3);
if (a != a2) {
    printf(2, "test single proc: failed. address of a2: %x, expected address: %x\n",
        a2, a);
}

// check ref count
for (i = 0; i < 8; i++) {
    if (shm_refcount(i) != 1) {
        printf(2, "test single proc: key=%d, count=%d, expected count=1\n", i, shm_refcount(i));
    }
}

stack_loc = ((uint)&a + SHM_SIZE - 1) & ~0xFFF;
free_space = (uint)h - stack_loc;
printf(1, "stack loc:%x, free space:%x\n", stack_loc, free_space);
h1 = (char*)malloc(free_space - 8); // should get all the remaining free mem
printf(1, "h1 mem: [%x, %x]\n", h1, &h1[free_space - 8]);
h2 = (char*)malloc(1);
if (h2 != 0) {
    printf(1, "test single proc: h3=%x, expected %x\n", h2, 0);
}
}

int
main(int argc, char *argv[])
{
    create_new_proc(test_single_proc);
    exit();
}

```

Step 8:

vi makefile.mk

add test\_shm after zombie in USER\_PROGS list of makefile.mk

```
# user programs
USER_PROGS := \
    cat\
    echo\
    forktest\
    grep\
    init\
    kill\
    ln\
    ls\
    mkdir\
    rm\
    sh\
    stressfs\
    tester\
    usertests\
    wc\
    zombie\
    test_shm\

USER_PROGS := $(addprefix user/, $(USER_PROGS))
```

Step 9:

\$ cd ..

Step 10:

\$make qemu-nx

\$ls

\$test\_shm

## OUTPUT

```
osd-190031187@team-osd:~/shared-memory-xv6
SeaBIOS (version 1.11.0-2.el7)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F94780+07ED4780 C980

Booting from Hard Disk..xv6...
lapicinit: 1 0xfee00000
cpul: starting
cpu0: starting
init: starting sh
190031187$ ls
.          1 1 512
..         1 1 512
README    2 2 1793
cat        2 3 9720
echo       2 4 9244
forktest  2 5 5996
grep       2 6 10864
init       2 7 9580
kill       2 8 9244
ln         2 9 9236
ls         2 10 10824
mkdir     2 11 9316
rm         2 12 9300
sh         2 13 16584
stressfs  2 14 9484
tester    2 15 9180
usertests 2 16 34888
wc         2 17 10060
zombie    2 18 9028
test_shm   2 19 12412
console   3 20 0
190031187$ test_shm
test_single_proc
shmgetat succeeds, a=9F000
stack loc:3000, free space:8F000
hl mem: [3008, 92000]
190031187$
```